

© 24m: 53s to test end





# ☆ Playlist

You love listening to music and just made a playlist of n songs on your MP3 Player named songs. Each song,  $s_i$ , in songs is ordered sequentially such that songs =  $[s_0, s_1, ..., s_{n-1}]$  and  $0 \le i < n$ .

The player has two buttons for switching between songs in the playlist,  $\uparrow$  (up) and  $\downarrow$  (down). If song  $s_i$  is currently playing, you can either press the  $\uparrow$  button once to switch to song  $s_{i-1}$ , or the  $\downarrow$  button once to switch to song  $s_{i+1}$ . If you press  $\downarrow$  while song  $s_{n-1}$  is playing, it will switch to song  $s_0$ . Similarly, if you press  $\uparrow$  while song  $s_0$  is playing, it will switch to song  $s_{n-1}$ .

You're currently listening to song  $s_k$  and decide you want to switch to some other song, q, in your playlist. What is the minimum number of button presses needed to switch from song  $s_k$  to song q?

Note: The same song may appear multiple times in the playlist. It is guaranteed that song q is in the playlist.

Complete the *playlist* function in your editor. It has 3 parameters:

- 1. An array of n strings, songs, where the value of each element  $s_i$  corresponds to the song at index i in the playlist (where  $0 \le i < n$ ).
- 2. An integer, k, the index of song  $s_k$ .
- 3. A string, q, the name of the song you wish to switch to.

It must return an integer denoting the minimum number of button presses needed to switch from song  $s_k$  to song q.

## **Input Format**

The locked stub code in your editor reads the following input from stdin and passes it to your function:

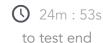
The first line contains an integer, n, denoting the size of songs (i.e., the number of songs in the list).

Each line *i* of the *n* subsequent lines (where  $0 \le i < n$ ) contains a string describing the value of song  $s_i$ .

The next line contains an integer, k, denoting the index of song  $s_k$ .

The next line contains a string, q, denoting the name of the song you must switch to.







- $0 \le k \le n-1$
- $1 \le length \ of \ s_i \le 100$
- 3
- It is guaranteed that song q is in the playlist.

### **Output Format**

- Your function must return an integer denoting the minimum number of button presses needed to switch from song  $s_k$  to song q. This is printed to stdout by the locked stub
- <sup>2</sup> code in your editor.

# 3 Sample Input 1

1

4

wheniseeyouagain borntorun nothingelsematters

cecelia

cecelia

## Sample Output 1

2

## **Explanation 1**

You start out listening to song  $s_{k=1}$  = "borntorun". By pressing the  $\downarrow$  button 2 times, you can reach  $s_3$  = "cecelia". Thus, we return the number of button clicks, 2, as our answer.

#### Sample Input 2

4
dancinginthedark
rio
liveoak
liveoak
0
liveoak

## Sample Output 2



O 24m: 53s to test end



### **Explanation 2**



You start out listening to song  $s_{k=0} = "dancinginthedark"$ . By pressing the  $\uparrow$  button 1 time, you can reach  $s_3 = "liveoak"$ . Observe that we could also have pressed the  $\downarrow$  button two times to switch to  $s_2 = "liveoak"$ , but two button presses would not be minimal. Thus, we return the number of button clicks, 1, as our answer.

2

#### YOUR ANSWER

3

4

We recommend you take a quick tour of our editor before you proceed. The timer will pause up to 90 seconds for the tour.

Start tour





O 24m: 53s to test end

```
static int playlist(String[] songs, int k, String q) {
                                       13 ▼
                                                                                                           if (songs[k].equals(q)) {
                                       14 	extstyle 	
                                                                                                                                 return 0;
                                       15
                                       16
                                                                                                           }
                                        17
1
                                       18 ▼
                                                                                                           for (int i = 1; i <= songs.length / 2; i++) {
                                                                                                                                  int up = (k - i + songs.length) % songs.length;
                                       19
                                                                                                                                  int down = (k + i) % songs.length;
                                       20
2
                                       21
                                       22
                                                                                                                                 if (songs[up].equals(q)) {
3
                                       23
                                                                                                                                                       return i;
                                        24
                                                                                                                                  }
                                       25 ₩
                                                                                                                                  if (songs[down].equals(q)) {
                                       26
                                                                                                                                                       return i;
                                       27
                                                                                                                                  }
                                       28
                                                                                                           }
                                       29
                                        30
                                                                                                          return -1;
                                       31
                                       32
                                                                                    }
                                       33
                                       34
                                                                                    public static void main(String[] args) throws IOException{
                                       35 ▶
                                                              ↔}
                                       70
                                                              }
                                                                                                                                                                                                                                                                                                                                                        Line: 9 Col: 1
```

Test against custom input

Run Code

Submit code & Continue

(You can submit any number of times)

Download sample test cases The input/output files have Unix line endings. Do not use Notepad to edit them on windows.

About Privacy Policy Terms of Service